Report sull’analisi del codice Assembly

Data 08/02/24

Introduzione

Il presente report si propone di fornire un'analisi dettagliata del codice Assembly fornito, senza previa conoscenza della sua funzionalità specifica. La traccia dell'esercizio richiede di determinare la funzionalità del programma in questione, basandosi esclusivamente sulle istruzioni e sul flusso di esecuzione del codice Assembly fornito

Spiegazione del codice

Il codice in questione fornito nella traccia di oggi è il seguente:

Iniziamo con una spiegazione riga per riga del codice:

- Push ebp: Questa istruzione spinge il valore corrente del registro base (EBP) nello stack. Questo è comunemente fatto all'inizio di una funzione per salvare lo stato del frame precedente.

- Mov ebp, esp: Questa istruzione imposta il registro base (EBP) con il valore dello stack pointer (ESP) corrente. Questo crea un nuovo frame di stack per la funzione corrente.

- Push ecx: Questa istruzione spinge il contenuto del registro ECX nello stack. ECX è uno dei registri general-purpose utilizzati per vari scopi, quindi qui viene salvato nello stack per essere utilizzato in seguito.

- Push 0 ; dwReserved: Questa istruzione spinge il valore 0 nello stack chiamato “dwReserved”. Questo valore viene utilizzato come argomento per una funzione successiva.

- Push 0 ; 1pdwFlags: Anche qui, viene spinto il valore 0 nello stack chiamato “1pdwFlags”. Questo è un altro argomento per la funzione successiva.

- Call ds:InternetGetConnectedState: Questa istruzione chiama una funzione denominata InternetGetConnectedState situata nella sezione dati (DS) del programma. Questa funzione è probabilmente utilizzata per controllare lo stato della connessione Internet.

- Mov [ebp+var_4], eax: Questa istruzione memorizza il valore di ritorno della funzione InternetGetConnectedState nella variabile locale [ebp+var_4].

- Cmp [ebp+var_4], 0: Questa istruzione confronta il valore memorizzato nella variabile locale [ebp+var_4] con 0.

- Jz short loc_40102B: Questa istruzione salta a loc_40102B se il confronto precedente ha dato esito zero, il che significa che non c'è connessione Internet disponibile.

- Push offset aSuccessInterne: Questa istruzione spinge l'indirizzo della stringa "Success: Internet Connection\n" nello stack. Questa stringa sarà probabilmente utilizzata per stampare un messaggio di successo.

- Call sub_40105F: Questa istruzione chiama una subroutine denominata sub_40105F, che probabilmente è responsabile della gestione della stringa e della sua visualizzazione.

- Add esp, 4: Questa istruzione aggiunge 4 al registro stack pointer (ESP), rimuovendo gli argomenti precedentemente spinti nello stack.

- Mov eax, 1: Questa istruzione imposta il registro EAX a 1.

- Jnp short loc_40103A: Questa istruzione salta a loc_40103A se non ci sono stati errori durante l'esecuzione dell'ultima istruzione.

Una delle richieste dell'esercizio riguarda la funzione "InternetGetConnectState", ci dice se la funzione in questione richiede 3 parametri per permettere al programma di creare una connessione ad internet.

Un breve cenno sulla funzione ci dice che; La funzione InternetGetConnectedState è una funzione API di Windows che controlla se il sistema ha accesso a Internet e, facoltativamente, se è possibile stabilire una connessione a una rete specifica.

Da una ricerca bibliografica online abbiamo scoperto che la funzione in questione in realtà richiede solo due parametri:

- Il primo parametro è un puntatore a una variabile DWORD denominato "1pdwFlags" che riceverà un valore che indica lo stato della connessione. Questo parametro è obbligatorio.

- Il secondo parametro è un valore DWORD opzionale denominato "dwReserved" che specifica il tipo di connessione da controllare. Se questo parametro è impostato su 0, la funzione controlla solo la connessione Internet generale. Se è impostato su 1, controlla se il sistema può stabilire una connessione a una rete specifica.

Nel codice fornitoci è stato assegnato il valore 0 ad entrambi i parametri con l'utilizzo della funzione " push 0 "

Basandoci sul codice assembly fornito e sulle istruzioni presenti, sembra che il programma stia effettivamente cercando di determinare se il sistema ha accesso a Internet. Ecco una spiegazione riguardo al funzionamento del codice:

Push ebp: Questa istruzione salva il valore corrente del registro ebp nello stack, in modo da poterlo ripristinare in seguito.

Mov ebp, esp: Questa istruzione imposta il registro ebp uguale al registro esp, creando così un nuovo frame di stack per la funzione.

Push ecx: Questa istruzione salva il valore corrente del registro ecx nello stack.

Push 0: Questa istruzione mette 0 nello stack. Questo valore è utilizzato come parametro dwReserved per la funzione InternetGetConnectedState, indicando che si desidera controllare solo la connessione Internet generale.

Push 0: Ancora una volta, 0 viene posto nello stack. Questo valore è utilizzato come primo parametro per la funzione InternetGetConnectedState, che riceverà il valore dello stato della connessione.

Call ds:InternetGetConnectedState: Questa istruzione chiama la funzione InternetGetConnectedState per controllare lo stato della connessione Internet.

Mov [ebp+var_4], eax: Questa istruzione memorizza il valore restituito dalla funzione InternetGetConnectedState nella variabile locale [ebp+var_4].

Cmp [ebp+var_4], 0: Questa istruzione confronta il valore memorizzato nella variabile locale [ebp+var_4] con 0, per verificare se la funzione ha restituito un valore che indica la presenza o l'assenza di una connessione Internet.

Jz short loc_40102B: Questa istruzione salta a loc_40102B se il confronto precedente ha dato esito negativo, il che indica che non c'è una connessione Internet.

Se il programma riesce a stabilire una connessione Internet, eseguirà il blocco di codice successivo che sembra gestire la situazione in cui c'è una connessione attiva.

Conclusioni

Grazie all'approfondita disamina delle istruzioni e della funzione chiave coinvolta, siamo in grado di confermare che il programma mira effettivamente a determinare se il sistema dispone di accesso a Internet. Questo tipo di analisi è essenziale per comprendere il comportamento e il funzionamento di applicazioni basate su Assembly e costituisce un passo fondamentale nella valutazione della sicurezza e dell'affidabilità del software.